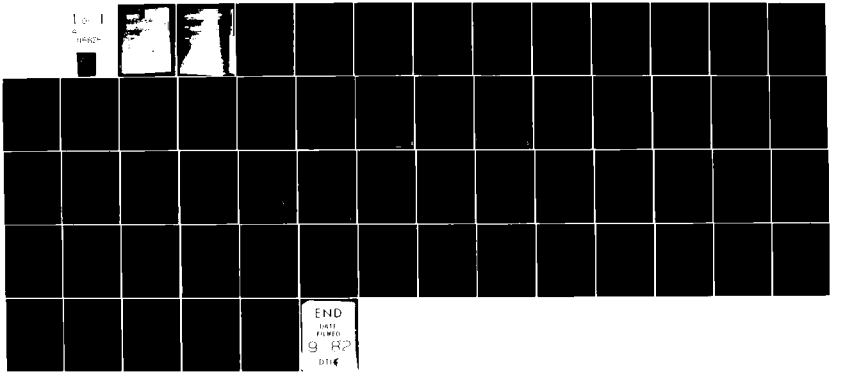
AD-A118 825 SOFTECH INC DAYTON OH F/G 9/2
JOVIAL J73 PROGRAMMING SUPPORT LIBRARY.(U)
JUN 82 A J CRUSCICKI, L SIMPSON, R SHEFFIELD F30602-80-C-0244
UNCLASSIFIED RADC-TR-82-162 NL



END
DATE FILMED
9 82
DTIC

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-82-162	2. GOVT ACCESSION NO. AD-A118825	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) JOVIAL J73 PROGRAMMING SUPPORT LIBRARY		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report July 80 - January 82
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Andrew J. Cruscicki, RADC Louis Simpson, Softech R. Sheffield, Softech		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0244
9. PERFORMING ORGANIZATION NAME AND ADDRESS Softech, Inc. FSD Dayton OH 45401		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 25320208
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE June 1982
		13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Andrew J. Cruscicki (COEE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Program Support Library	Computer Software
Configuration Control & Management	Software Documentation
Software Development Environment	Top-down Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The JOVIAL J73 Programming Support Library (J73 PSL), an environment for the orderly creation of J73 Software Systems, is presented. This technical report has been developed as a self-contained/stand-alone document. That is, the TR first presents a thorough contract overview followed by a discussion of the technological concepts (i.e., Top-down programming and segmentation) incorporated as features of the J73PSL. A brief command verb summary then proceeds a section devoted to the relationship between the J73PSL and configuration control and management. The remainder of the TR is devoted to certain project tasks (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

followed by a discussion of perceived future directions for this technology.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 BACKGROUND AND INTRODUCTION

The JOVIAL J73 Programming Support Library (J73 PSL) was a fixed price level of effort acquisition by the Rome Air Development Center (RADC) of the Air Force to SofTech's Federal Systems Division requiring system design, development, test and maintenance within 1.5 years of contract award. It included a state-of-the-art PSL study, software design, development and test, user training, maintenance, and enhancements.

The J73 PSL provides configuration control for development and maintenance of JOVIAL J73 software systems. Storage is provided for source code, statistics, documentation, design, object modules, load modules, symbol tables, and listings. Within the J73 PSL the user can create, modify, compile, link, and execute code. Statistical data is automatically collected and maintained in the database. Reports are generated which analyze the statistical data on a segment, module, program, or entire library basis. Seven levels of configuration control are provided, in which parallel datasets of each of the types (source, object, documentation, etc.) are stored. Authorization to read and write at the different configuration levels is strictly controlled. Interfaces to J73 compilers, support compilers (FORTRAN, host assembler), and other software tools are provided.

The project began with a study of existing Program Support Libraries. SofTech subcontracted part of this study to General Research Corporation of Santa Barbara, California. The results of the study were included in the PSL Interim Technical Report and summarized below. Of the seven studied, the PAVE PAWS PSL was chosen as a design baseline. It was believed that direct translation of much of the PAVE PAWS PSL code, even though it was hosted on a different computer and written in a different dialect of JOVIAL, would be possible.

A Functional Description was written, in which the requirements of the Statement of Work were matched to PAVE PAWS PSL directives, newly designed directives, or to database design. An efficient implementation of the requirement for a hierarchy of seven library levels to hold a number of different kinds of data on each level was chosen. Although the PAVE PAWS PSL proved to be a good design baseline, direct translation proved to be an obstacle and in the later stages the project became a straightforward software development effort.

A System Subsystem Specification was published, in which the design details of the main J73 PSL program were delineated. While this design, with some revisions, was implemented in the PSL, five additional independent programs were also written to carry out some functions during independent batch execution. A major feature of the J73 PSL is the spawning of batch background jobs--to provide a standard interface to run compilers, linkers, etc--and these additional programs were necessary to process data in conjunction with these batch background jobs. The PSL as implemented consists of four independent JOVIAL J73 programs, the main PSL program comprising about 95% of the code. In addition, two short FORTRAN programs are used to route listings. The main PSL program, called PSLMAIN, consists of 81 independently compilable modules, of which 65 are written in JOVIAL J73, eight in FORTRAN, and eight in IBM assembler. FORTRAN and assembler routines primarily serve to accomplish I/O operations, since JOVIAL J73 has no I/O in the language. Section 4 (Detailed Design) briefly describes each program in the PSL system and each module in the largest program of the system, PSLMAIN.

Section 5 (Testing) describes three phases of testing--unit, integration, and qualification--and the testing criteria which were common to all three. Common testing criteria included testing both legal and illegal inputs, checking correct program execution, and testing capacities and boundary conditions. In addition, integration testing, which involves the first attempt to operate the developing system as a whole, uncovered interfacing problems. Qualification testing systematically tests each function of the system as it relates to requirements in the Statement of Work. Techniques include visual inspection of code and documentation, operation of the PSL in both batch and interactive modes, and use of the system commands to look at the results of the PSL from outside the PSL.

Section 6 (User Indoctrination) describes the training class held at ASD on October 20-24, 1981. The first two days were lecture and the last two were primarily laboratory. The first laboratory day was devoted to program development using a preexisting PSL database, while the second laboratory emphasized the management aspects of the PSL.

Section 7 enumerates maintenance tasks and possible enhancements which could be negotiated. Tasks to be completed during maintenance include publishing the Program Specification and the Program Maintenance Manual, updating documentation which goes into the former, delivering the PSL software, and dealing with numerous small but important problems which still remain. Possible enhancements include a directory command, a software problem report (SPR) generator, and a requirements traceability matrix generator.

1.1 J73 PSL Software Design Requirements.

The PSL Statement of Work (SOW) describes requirements which must be met by the J73 PSL. These requirements are intended to meet the JOVIAL J73 programming support needs through the introduction of a modern software engineering tool. The SOW requirements and the J73 PSL's specific implementation of them are discussed in the following paragraphs.

The PSL must be written in JOVIAL J73, according to MIL-STD-1589B. Exceptions could be made for significant reasons, but any exception had to be approved by the sponsoring agency. Since JOVIAL J73 has no input/output, the I/O routines were written in FORTRAN and assembler.

The PSL must provide seven library levels which are structured in a vertical hierarchy. As code matures, the user shall have the capability of transmitting the code to a higher library level. The user will be able to transmit modules, which consist of all source segments which make up a J73 module, plus the associated text and statistics segments, to a higher level in the PSL by the XMIT directive. The seven levels provide a structure for configuration control of a developing and/or operational software system. Each library level has usage

conventions which limit what actions may be performed on code. These usage conventions are established by the program manager at project set-up time. The lowest level could represent the most unrestricted area and the highest level could be the most restricted. Typically, programmers are allowed access to the lowest levels and the program manager to the highest levels where the released software resides.

The SOW requires an automatic drawdown feature to be used in the transmission of code to a higher level. Automatic drawdown means that a segment group is copied from a higher to a lower level when the need arises. If not all the segment groups which comprise a module exist on the level from which the user wishes to transmit the module, the missing segment group will be copied to the lower level at that time by automatic drawdown.

The PSL must provide storage capability for source modules, textual data, and object modules. Each PSL library level will have these types of segments as well as other types as shown in figure 1-1. A J73 source code module may be one segment or a group of segments that are related by !COPY statements in the code. Source segments may also contain source code for other languages: FORTRAN, PDL (Program Design Language), or assembler. The !COPY keyword may also appear in source segments of these other languages; it will serve here also, as in the case for J73 source code, to provide for combination of segments into a module. Data for input to a program will be stored as a source segment, as will linker control commands. Textual data, intended to be documentation for a source segment, will be stored as a segment of type TEXT. Each segment of type SOURCE will have a TEXT segment associated with it. Object modules will be stored as segments of type OBJECT. There will also be segments of type LIST (for compiler, assembler, or other output), LOAD (for program execution code), and SYMTAB (for symbol tables produced during compilation of J73 source modules). Segment groups may only be created by the ADD directive or as stubs due to a !COPY string in the source segment. Thus top-down design is facilitated.

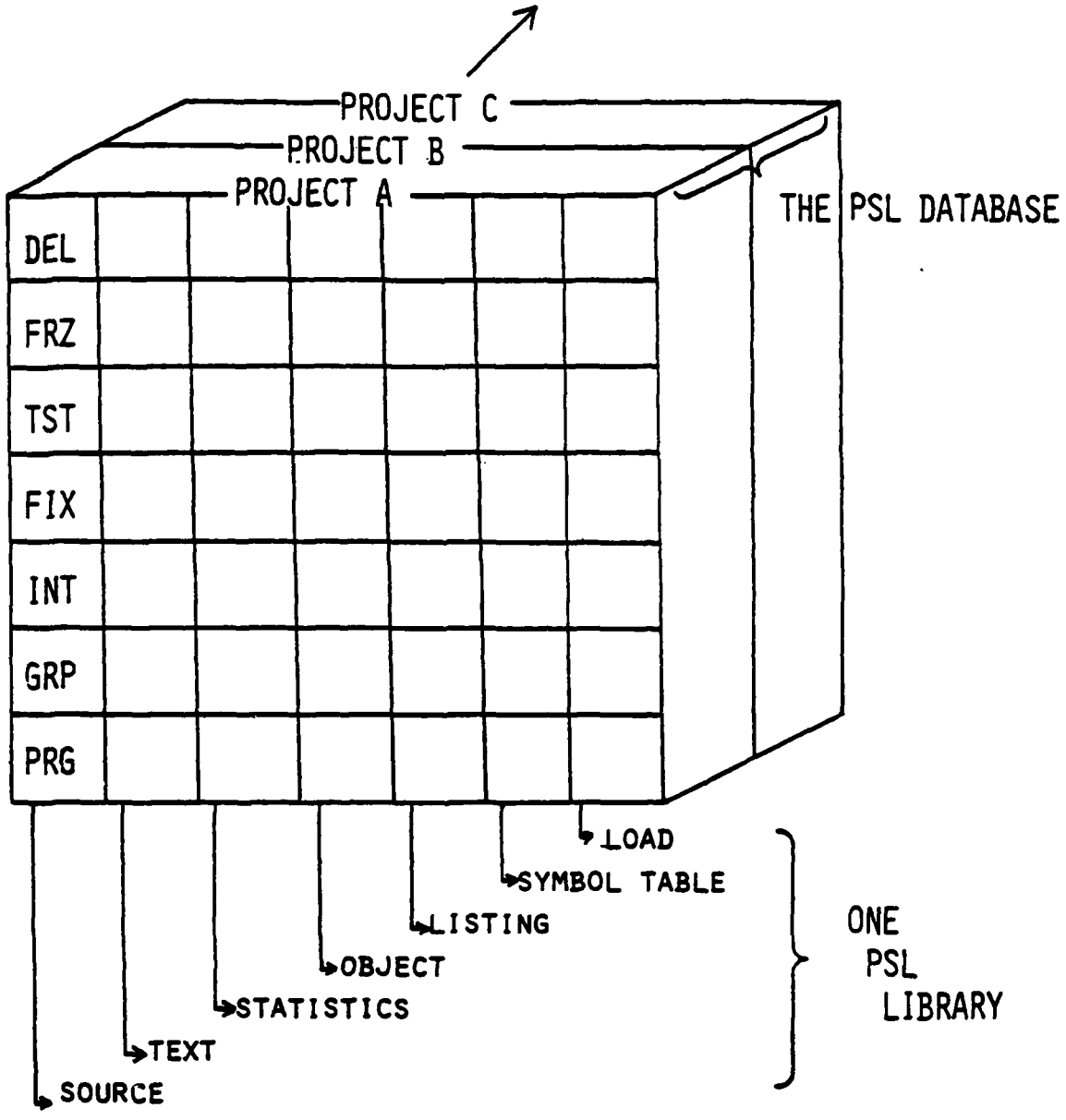


Figure 1-1. J73 PSL Database Structure

The J73 PSL must permit the user to name and rename the keywords and usage conventions associated with each library level. There must be a way to define user restrictions on a PSL verb basis at the various library levels. The PSL implements these requirements in the following manner. After initializing a PSL library with the NEWLIB directive, which establishes the user as the manager of that library, the manager may set up the library level keywords and usage conventions by modifying two special segments. The segment AUTHORIZATIONS will contain information in a specific format which establishes the USERIDs to be associated with a number of usage groups. The manager defines for each usage group the access levels for each directive verb. The segment KEYWORDS defines the names of the seven library levels and also establishes keywords to be employed in TEXT segments. The manager may redefine user restrictions, level keywords, and text keywords by modifying the AUTHORIZATIONS and KEYWORDS segments.

The PSL is required to perform multi-library-level searches for the purpose of compilation or for linking loadable object code. The implementation of the PSL will combine source segments into modules during processing of the !COPY keywords in the source. The linker, when invoked by the LOAD directive, will do an automatic search of all object segments in the designated PSL library.

The J73 PSL must interface with the JOVIAL J73 compiler and at least five additional compilers. The PSL directive, COMPILE, invokes any one of the following: the default J73 compiler, the host FORTRAN compiler, or the host assembler. Which of the three is invoked is determined by the language of the source module being compiled. Options for a compiler or assembler may be specified by using the options parameters. Other versions of compilers, or compilers for other languages may be invoked through use of the PERFORM directive, provided their interfaces have been implemented. An interface consists of code which modifies a job control statement and sends out a spawned job. The PERFORM directive has OPTION parameters which serve the same purpose as the OPTION parameters of COMPILE.

The PSL must provide seven keywords to describe the program type. The following keywords for source segment type are provided: MAIN, PROC, COMPOOL, DATA, COPY, LOCAL, and SUBR.

The J73 PSL must provide an accounting record to store statistical data for report generation. The PSL will automatically collect statistics during execution of the following directives: ADD, MODIFY, COMPILE, and LOAD. A statistics segment is associated with each source segment. The user has no direct access to the statistics segments. These segments are generated automatically and are only readable in the form of a management report generated by the REPORT directive. In the REPORT directive, the user may specify whether statistics from all levels or a range of levels are to be used. Four types of reports are possible: segment statistics summary, module statistics summary, program statistics summary, and library summary. In the first three types, the user may specify a name or receive a summary of the statistics of all segments, modules, or programs. These reports may also be limited to a range of library levels. The library summary consists of a two-page report summarizing the contents of the library in use.

The SOW requires that the PSL be as machine-independent as possible, with concentration of machine dependent software, minimizing the effort to rehost the PSL. Machine dependent code in the implementation of the PSL will be mostly isolated in routines for input/output management and tool interfaces. Interfaces for compilers and external tools consist of datasets containing job control language to be used for spawned job generation and routines which modify these interface datasets.

The J73 PSL must be independent of external software systems, such as data base management systems. No external software systems are used by this implementation.

There should be minimal revision of machine dependent parameters required in order to rehost the J73 PSL. The isolation of machine dependent parameters to a single compool or a few compools will facilitate rehosting.

There should be a capability to allow interface with non-PSL external software testing tools, such as a code analyzer. The user should have the option to store the results of the execution of the tool within the PSL. External tools, like the non-default compilers and compiler versions, may be interfaced through the PERFORM command. The user may enter the results of the execution into the PSL with the IMPORT directive. Options available for the external tool may be specified using the OPTION parameters.

Text segments should be up to three pages long, at fifty lines of standard card image format. There should be a text segment associated with each source segment. The source and text segments should have the same name. Transmission between library levels should move the source and text segments together.

Editing of source and text segments should be possible using the host text editor. The user may access copies of PSL database segments which have been created in a special partitioned data set by the EXPORT command. Later the host editor-modified copy of the text or source segment may be returned to the PSL database by the IMPORT command. There will be no statistical update of IMPORTed segments because there is no assurance that the replacement text or source is what it claims to be. Within PSL, there is a modify directive which can be used to modify PSL segments, but is not at this time of comparable power with the host text editors. Statistics are updated when the MODIFY directive is used.

The PSL must provide for the definition of twenty keywords to be used with TEXT segments. Each keyword must contain a maximum of fifteen alphanumeric characters. Keywords are defined by the manager when he creates and modifies the KEYWORDs special segment. A specific format within this special segment defines the character strings which are the keywords. The manager may redefine the keywords by MODIFYing the KEYWORDs segment.

The PSL must provide a command to allow the user to search TEXT segments throughout the PSL library for the occurrence of a keyword

and print out the results of that search. Upon a successful search for the primary keyword, the data associated with secondary keywords would be printed out. The implementation of the SEARCH directive provides the user with a number of options. The search may be conducted for all text segments in a PSL library, only the text segments on a given level, or only the text segment associated with a given source segment. A successful search can require not only that the primary keyword be found, but that a character string occurring as a parameter in the directive be in the text associated with the primary keyword. If secondary keywords are included as parameters in the directive, the text associated with the secondary keywords is printed out if the search is successful.

A document function should be provided in the PSL which allows the user to print textual information stored at any of the seven library levels. The user may specify the following options: name(s) of the text; library level(s); selection of data to be printed based on keywords; up to three header lines; whether the header appears at the top of the first page or at the top of every page; selection of page numbering range; selection of maximum number of lines per page (default fifty including headers and blank lines); and selection, by line number, of blank lines. The implementation uses the LIST directive. Besides listing text segments, the LIST directive will also permit listing of SOURCE and LISTING segments. The specific formatting for TEXT and SOURCE segments will be controlled by subdirectives. A parameter of the LIST directive is used to designate the type of segment being listed. In addition to segment (SOURCE or TEXT), module (all source segments associated by !COPYs to makeup a source module), compile listing, load listing, and PERFORM listing, there is also a capability for listing a hierarchy of segments in a module.

The PSL must operate in both batch and interactive modes. All capabilities should be available in each. In interactive mode, the user executes the PSL interactively at a terminal. This means that the PSL program executes time slices during the session and the user enters

directives and inputs to it at the terminal. Whenever a directive and its associated parameters and keywords are input at the terminal, execution of the directive occurs immediately following input of this set of data; a response, the JOBLOG, then appears at the terminal. In batch mode, the user commands are placed in a file before the batch PSL job is submitted.

The user should be able to display stored source and text data on a terminal, modify the data, and restore the modified version within the PSL. Display is accomplished by the LIST directive, with the listing routed to the terminal. Modification can be done with the MODIFY directive, which also restores the modified source or text. Alternatively, the user may modify EXPORTed data using the available text editors while outside the PSL and restore it to the PSL data base with the IMPORT directive.

The user should be able to create and execute job streams, such as a job stream to compile a source module, accessing source segments and SYMTABs which may reside at various levels within a PSL library. The user will accomplish this by using the COMPILE, LOAD, and PERFORM directives, which function as discussed above.

With these explicitly stated PSL design requirements, a study of available state-of-the-art programming support libraries would be necessary. SofTech subcontracted to the General Research Corporation of Santa Barbara, California for this part of study effort. A general description of each PSL system studied will be discussed in the next section.

1.2 Study of PSL-like Systems

A study was made of PSL-like systems to investigate the possible use of a single baseline for the J73 PSL and to identify techniques and features which may be included from other successful PSL implementations. The PSL-like systems were:

a. PAVE PAWS PSL, hosted on a CYBER 175.

b. Defense Mapping Agency (DMA) PSL, hosted on a Sperry Univac 1100 series machine.

c. Microprocessor Software Engineering Facility (MSEF), hosted on a Digital DECsystem-10.

d. AFAL Software Development and Verification System (SDVS), hosted on a Digital DECsystem-10.

e. IBM Structured Programming Facility, hosted on IBM-370 or equivalent machine.

f. The EXEC 8 OS hosted on a Sperry Univac 1100.

g. Bell Laboratories Programmers Work Bench (PWB) hosted on a Digital PDP-11/70.

1.2.1 PAVE PAWS PSL.

The PAVE PAWS is a fixed base Phased Array Warning System utilized for the detection and attack characterization of Submarine Launched Ballistic Missiles (SLBMs) which penetrate the radar coverage. The project produced approximately 211,000 lines of software, about 17,000 of which were devoted to PAVE PAWS PSL.

The PAVE PAWS PSL provided the PAVE PAWS development program with the capabilities of configuration control, support software tool invocation and management reporting in a batch operation environment. Configuration management occurs in many forms, like the multi-level library environment with command verbs supplied for library content manipulation. A language preprocessor is supplied to add structured programming constructs, copy segment processing, and program hierarchy generation. Tools invoked by the PAVE PAWS PSL are limited to the J3 Compiler, COMPASS ASSEMBLER, and CDC linkage editors. Management reports are based on statistical data generated for PAVE PAWS libraries, programs or segments. The PSL control (control software) and PSL functions (verb function) together comprise the module referred to as the PAVE PAWS PSL. In addition, there are two other modules which, while operationally separate from the PSL, are conceptually part of it. The Language Pre-Compiler (LPC) is responsible for assembling an entire program from the set of modules in it. The Management REPorting (MREP) program is responsible for the producing all PSL reports.

1.2.2 Description of the Microprocessor Software Engineering Facility (MSEF)

MSEF is written in the programming language "C" and hosted on a PDP-11/70 under the UNIX operating system. The MSEF provides UNIX hosted support for the storage and organization of the components of large systems of computer software. The MSEF assists project management, configuration control, programming, implementation, testing, and in-service upgrade of such systems. The MSEF acts as the framework to support software development tools in a unified and coordinated environment.

In practice, systems are built of many subsystems which, in turn, are combinations of subroutines and other subsystems. Such multi-level structuring helps to simplify and organize the overall project. This multi-level (hierarchical) structure is also useful for organizing ancillary information such as resource budgets, documentation, test results, test scripts, and expected test results.

1.2.3 Software Design and Verification System (SDVS)

The Software Design Verification System (SDVS) is a support software package specifically intended for testing and maintaining operational software. SDVS provides simulation capabilities and allows the user to generate tests to exercise the operational software. About 20% of SDVS is devoted to Program Support Library functions, chiefly configuration control and text editing.

In its implementation, each software project assigns a person to be SDVS data base manager. This person assigns space and names of all modules to be used in the project. Files are source, object, load, etc. with appropriate use conventions for each type of the file. There is no allowance for text files *per se*. A specification file may be created by the data base manager which will hold high level design descriptions. These files are accessible only by the data base manager and are therefore not text files containing a PDL or other detailed design description which can be created and moved with the code files.

All files and updates of files are stored indefinitely. Baselines, called versions, are periodically established. Updates are stored in the form of difference files, called revisions. While a hierarchy is implicit in the ascending version and revision numbers, no difference in access or use is inherent in SDVS.

Statistics on file access and usage are automatically collected. The statistics collected are basic information on the time of day, user identification and function performed (e.g., compile, link, etc.). These statistics are collected and put into a single repository file or data dump for every user within a project. Only the project SDVS data base manager has access to this dump.

SDVS supports testing at various levels. Simulation languages were created for instruction level (for a specific machine) and statement level (for HOL implementations) simulations. The statement level simulation created a series of commands which could be used to initialize and activate collection of various forms of data. Conversational language commands were also supplied to begin and end program execution as well as provide conventional commands (IF, WHILE, etc.) to build strings of test commands. Special commands were added to interact with environment simulations so that test cases could be suspended and rerun for verification of test results. These simulations are used to do module test on host computers. Little statistical data is kept on these module tests other than the user information contained in the data base manager repository file. Aircraft simulations are also supplied under SDVS with the capability of creating a flight scenario and recording data in flight code modules as well as aircraft simulation. Portions of the flight scenario can be recreated and rerun to verify data of interest. SDVS also provides standard statistical packages, as well as the ability to create special purpose diagnostic packages, to analyze recorded data. Although the SDVS implementation collected data, no routines were ever agreed upon to do statistical analysis nor limit set on the number and type of data items to be collected.

1.2.4 IBM Structured Programming Facility (SPF)

The IBM is not a PSL, but provides an extremely convenient method for file editing and job entry into the host OS. It does not include any library control other than that inherent in the host OS file structure.

In short, SPF is totally an IBM host OS-dependent system. It was designed to facilitate IBM job entry by minimizing the typical IBM user-hostile interface as well as "...to take advantage of the characteristics of IBM 3270 display terminals..."

1.2.5 UNIX Programmer's Workbench (PWB)

UNIX/PWB is a system designed and implemented by BELL labs for in-house work on DEC computers. It has found wide user acceptance for several reasons.

The UNIX/PWB provides a highly interactive user interface. The interactive text editors allow for manipulation of both source and textual data. Multiple files can be displayed side by side on a CRT. The system is rich in file manipulation commands and the user is also able to create job streams.

There are also specific features integral to the UNIX design. A separate configuration control tool is available with an accounting directory containing problem reports. An unlimited hierarchy is available with levels based on code maturity. The UNIX/PWB provides support to various compiler versions (e.g., C, ASM, FORTRAN, COBOL, and JOVIAL). Separate file name delegation for object (OBJ), execution (EXE) and documentation (TEXT) are provided and a search of multiple files and a listing of the output results are also available.

1.2.6 EXEC 8 Operating System

EXEC 8 was included in the study because it had been claimed to satisfy most of the requirements for DMA PSL. EXEC 8 does not meet many of the requirements of the J73 PSL yet it is very well received by the user of this system. It is an interactive system with a

reasonable text editor which, while not as good as UNIX/PWB or the SPF screen editors, is far better than the usual batch oriented editor. It is possible to manipulate files with simple commands, to group associated files together, and to create job stream files. Although EXEC 8 is not a PSL but an operating system, it is easy to see how a machine-dependent PSL could be constructed using UNIVAC system functions.

EXEC 8 does contain a file system which is easily stored, accessed and modified. EXEC 8 also has a reasonable text editor and a method of saving incremental changes.

EXEC 8 does not save user oriented statistics, management data or source-associated text. It does not provide data security, keyword processing, multi-level searches or text formatting. The UNIVAC EXEC 8 contains a four-level hierarchy. The first level is described by a qualifier which is usually the user job name, the second level is described by a file name, the third level is the cycle or version, and the fourth is the element. One paper suggested the qualifier could be the project identifier and the cycle be used for version control. Indeed, the file names could be assigned to different maturity stages and the elements could contain the different types of source, object, text, or data.

2.0 JOVIAL J73 PSL PROGRAMMING ENVIRONMENT

The JOVIAL J73 Programming Support Library (J73 PSL) is designed to promote modern software engineering and management practices for J73 computer program developments. The PSL enhances software development through the capabilities of:

a. Automated Configuration Control and Management
b. Invocation of J73 Development and Test Support Software
c. Enforcement of Top-Down Design and Development Through Program Segmentation.
d. Seven Level Hierarchy of Code Development Maturity
e. Module and Hierarchy Listings
f. File and Command Authorization Checking
g. Multi-Level Management Reports on Program Development Status

2.1 Top-Down Programming and Segmentation

Top-Down programming is a method of designing (and implementing) software through developing top level functions first. The details of the program materialize through successive iterations of the problem. For each iteration, known functions or newly specified subfunctions are continually integrated into the program. One objective of this method is to purposely break the program into readable blocks of code that should not exceed a standard page length. Thumbing through multiple pages of code for a single function is thus avoided and efficiency is achieved. Program functions and subfunctions are abstractly viewed as "black boxes." As program development proceeds, these "boxes" are coded. The J73 PSL allows the user to perform Top-Down Design and Development through segmentation. For a given main program one and only one top level segment is allowable. Lower level segments

which identify either a function or subfunction are placed in the top level segment by a !COPY. The PSL automatically creates stubs for these lower level functions. A stub can be simply viewed as a "black box." The J73 PSL automatically creates a stub for ADD, IMPORT, and MODIFY commands. A stub initially consists of a source segment, a text segment, and a statistics segment.

2.2 Module and Hierarchy Listings

The primary difference which distinguishes top-down structured programming from traditional programming methods is the ability to generate and work with the program's logic structure. In this respect, delegation of subprograms, etc. to programming team members is possible. The J73 PSL provides LISTings of several types of which directly foster top-down design and implementation. A listing of the type HIERARCHY supplies an indented copy tree for a module. This is not a call tree, since only COPY segments are in it (FIGURE 2-1). The J73 PSL also provides listings for specific segments and modules, as well as output listings from compilers, assemblers, linkers, and user-program execution. A module listing is a depth-first order listing of all segments (main and copy) in a module.

2.3 Hierarchical Library

The most noteworthy aspect of the J73 PSL is the feature of automated configuration control and management. Controls on the progression of software are at the discretion of a single person, usually the computer program manager. In this manner, program development, test, and integration proceeds in a controlled and orderly fashion. The seven-level hierarchical library structure is the primary vehicle for code progression in the J73 PSL. Software segments are entered into the PSL library using a user-specified name and user-specified level. Each level in the PSL is separate and distinct, yet a given segment may reside at multiple levels. For example, a segment may reside at both levels four and five. Level five could be the qualification test level and level four the "fix" level. Minor changes to the code could

```
Module:  PSL'ONE                                    LEVEL:  ONE
PSL'ONE                                                     ONE
     PSL'TWO                                                ONE
     PSL'THREE                                              TWO
         PSL'FOUR                                           TWO
     PSL'FIVE                                               ONE
         PSL'SIX                                            FOUR
             PSL'SEVEN                                      SIX
```

Figure 2-1 Hierarchy Listing

be performed at level four for code undergoing qualification testing. Thus, to completely identify a segment in the PSL library, both the segment longname and level must be specified. This provides a simple mechanism for parallelism in development, error correction, and version modification. The levels and usage conventions for the PSL library are shown in Figure 2-2. Important to this configuration hierarchy is the need to move code from one level to another. A program element is ready to change control level when it has satisfied predefined qualification criteria and is ready to be placed under more stringent change control. The XMIT directive effects code movement from one level to a higher level. XMIT will use the "drawdown" feature of the PSL to construct the entire source module hierarchy. It will then move all those segments up to the specified "to" level. The "automatic drawdown" feature allows library operations to be addressed to a specific library level and if the element does not exist at that level, successively higher levels will be searched until the element is found. Once the element is found it will be treated as if it were found at the originally requested level. This is based upon the upward migration of software through library levels and the recognition that all elements above the requested level have already satisfied the functional benchmark associated with that level.

2.4 PSL Authorization Checking

The hierarchical nature of the PSL library system readily lends itself to the systematic application of change control procedures. Since the migration of programs from level to level requires that more stringent benchmarks have been satisfied, the software stability (and the corresponding authorization required to effect change) continually increases from the lowest level to the highest. This is addressed in the PSL through an authorization verification scheme which recognizes the user-ID and restricts the operations and the library levels which they may use. This scheme is based upon a combination of user identity and organization and it disallows:

a. Operations on software which is not in the province of the

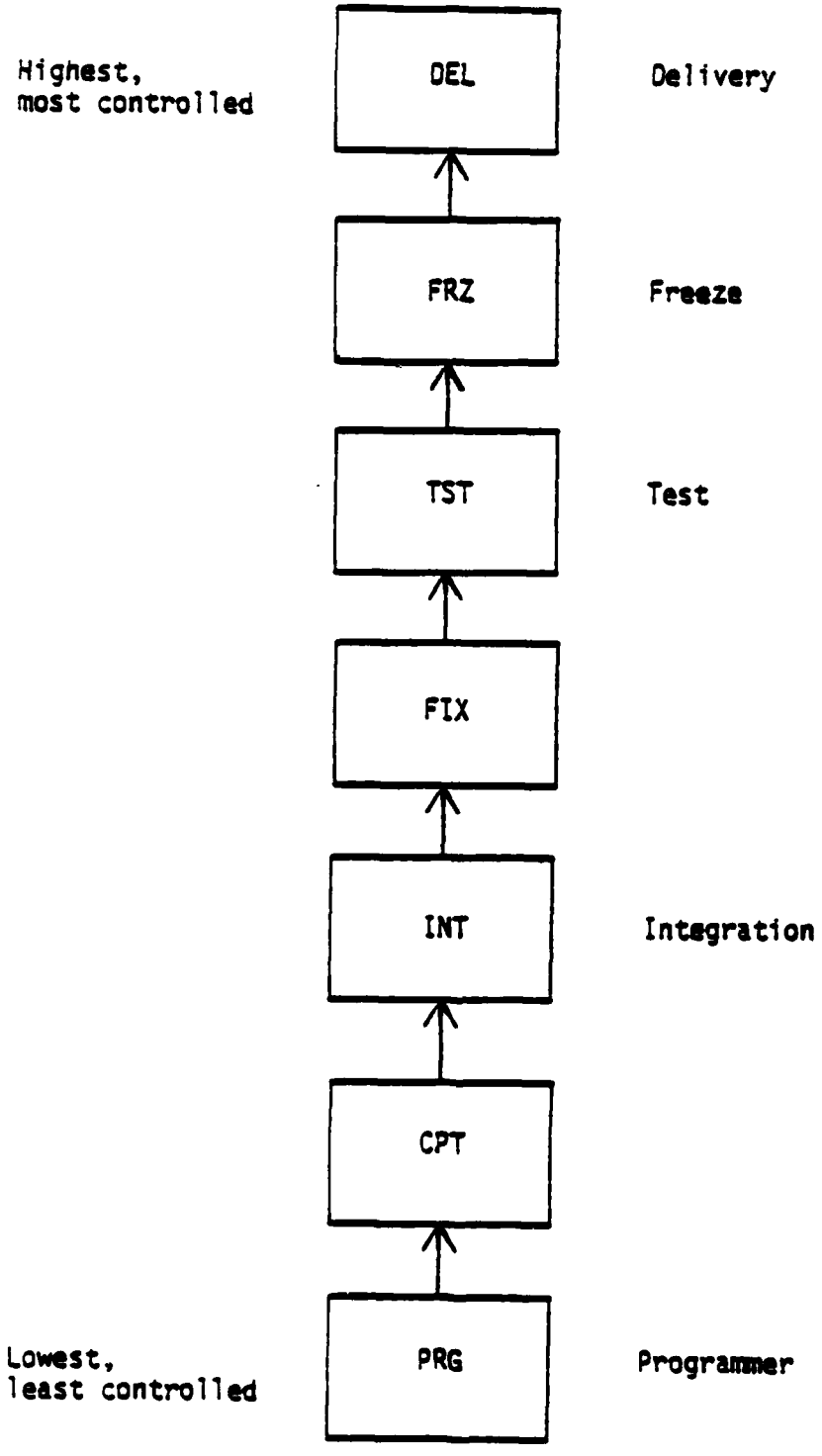


Figure 2-2 PSL Levels

organization.

b. Transactions at library levels at which the user is not authorized.

c. Execution of special PSL verbs for which the user is not authorized.

Among other things, implementation of this authorization check may prevent a programmer in one department from changing code belonging to another department, inhibit the Development organization from making changes to software which has been delivered to Test, prevent Test from accessing any software which has not been delivered to them, and disallow any source change activity (ADD, MODIFY) above the INT level of the library.

2.5 Management Statistics Reporting

The J73 PSL maintains statistical data for each segment, module, and library. Segment data is derived from the user-specified values when the segment was ADDed (longname, shortname, language, segment type, etc.) or computed automatically by the PSL (creation date, data and time of last change, number of lines and user identification).

Four types of reports can be produced with the REPORT directive: SEGMENT, MODULE, PROGRAM, and LIBRARY. Statistics pertaining to individual segments, modules as compiled, programs as linked, or the PSL library as a whole are used to generate the respective four kinds of reports. Figures 2-3 through 2-6 illustrate the four report types.

```
19 OCT 1981 18:00:25.55          J73 PSL VERSION 81.10.20          SUMMARY BY SEGMENT          PAGE: 1
        CPCS = PSL'REPORT2                                  LEVEL RANGE: TWO          TO SEVEN

- SEGMENT NAME                                  LANGUAGE   UNIT TYPE  AUTHOR
        LEVEL          STATUS   DATE/TIME CREATED      MET SZ ORG SZ  LAST DATE/TIME CHANGED  VN ED TOT CHGS CHGS/VN LAST CHGR

PSL'REPORT2                                     JOVIAL     MAIN       SFTRJS
        TWO            REAL   19 OCT 1981 17:54:26.38     15    15  19 OCT 1981 17:54:26.38  A0  0     0        0    SFTRJS
PSL'REPORT2                                     JOVIAL     MAIN       SFTRJS
        THREE          STUB   19 OCT 1981 17:48:16.93      0     0  19 OCT 1981 17:48:16.93  A0  0     0        0    SFTRJS
PSL'REPORT2                                     JOVIAL     MAIN       SFTRJS
        FOUR           REAL   19 OCT 1981 17:53:30.77      0     0  19 OCT 1981 17:53:30.77  A0  0     0        0    SFTRJS
PSL'REPORT2                                     JOVIAL     MAIN       SFTRJS
        SIX            REAL   19 OCT 1981 17:52:43.00      4     4  19 OCT 1981 17:52:43.00  A0  0     0        0    SFTRJS
PSL'REPORT2                                     JOVIAL     MAIN       SFTRJS
        SEVEN          REAL   19 OCT 1981 17:52:06.45      0     0  19 OCT 1981 17:52:06.45  A0  0     0        0    SFTRJS

        TOTALS   TOTAL NUMBER SEGMENTS   TOTAL NUMBER STUBS   TOTAL MET SZ   TOTAL ORG SZ   TOTAL NUMBER CHANGES   AVG LINES/ SEGMENT

                              5                    1              35             35                     0                   7
```

Figure 2-3 REPORT SEGMENTS Summary

```
08 DEC 1981 15:28:08.52          J73 PSL VERSION 01.10.20          SUMMARY BY MODULE          PAGE: 1
        CPCB = PSL'ONE                                    LEVEL RANGE: ONE          TO SEVEN

MODULE NAME                                    LEVEL
        TOT LINES NO SEGS NO STUBS  LAST DATE/TIME CHANGED  LAST DATE/TIME COMPILED  TOT STATEMENTS COMP ATTEMPTS COMP SUCCESS

PSL'ONE                                        ONE
            37        4        0   09 OCT 1981 08:21:34.64  25 NOV 1981 09:25:10.57        35            24          YES
PSL'ONE                                        SEVEN
            16        4        0   09 OCT 1981 08:21:34.64  23 OCT 1981 10:59:27.71        15             5          YES

        TOTALS  TOTAL NUMBER  TOTAL NUMBER  TOTAL NUMBER  TOTAL NUMBER  TOTAL NUMBER  AVERAGE LINES/
                   MODULES       LINES       SEGMENTS       STUBS       STATEMENTS       MODULE

                       2            53             0             0            50            24
```

Figure 2-4 REPORT MODULES Summary

```
19 OCT 1981 17:40:43.23          J73 PSL VERSION 01.10.20          SUMMARY BY PROGRAM          PAGE: 1
        CPCB = PSL'ONE                                    LEVEL RANGE: ONE          TO ONE

PROGRAM NAME                                   LAST DATE/TIME LOAD      LOAD SIZE     LOAD SUCCESS

PSL'ONE                                        19 OCT 1981 13:27:05.34      7618          YES

        TOTALS  TOTAL NUMBER  TOTAL LOAD
                  PROGRAMS       SIZE

                       1          7618
```

Figure 2-5 REPORT PROGRAMS Summary

81/01/25 15.13.14 J73 PSL VERSION 81/01/01. SUMMARY BY LIBRARY

LIBRARY LEVEL - PRG

--LANGUAGE--	--SEGMENTS--	--STUBS--	--MODULES--	--PROGRAMS--	--LINES--	--CHANGES
JOV	725	97	450	143	23484	4585
FORT	116	13	36	36	3813	549
ASM	33	0	11	11	445	79
OTHR	28	1	26	26	396	9

: (lines omitted)

LIBRARY LEVEL - FRZ

--LANGUAGE--	--SEGMENTS--	--STUBS--	--MODULES--	--PROGRAMS--	--LINES--	--CHANGES--
JOV	1271	6	900	22	34764	4727
FORT	522	0	84	84	15172	1976
ASM	6	0	2	2	145	38

Figure 2-6 Second Table in REPORT LIBRARY Summary

2.6 J73 PSL Directives

This section provides a brief description of each of the PSL directives.

2.6.1 ADD

The ADD directive adds a new segment group to the library, consisting of source, text, and statistics segments. Stubs are generated for all !COPY segments in the added source segment.

2.6.2 CHECKPOINT

The CHECKPOINT directive creates a copy on tape of every file in the PSL library being used.

2.6.3 COMPILE

The COMPILE directive spawns a batch job stream which invokes, according to the language parameter in the segment's statistics, either the default JOVIAL J73 compiler, the default FORTRAN compiler, or the host assembler. The segment identified by <longname> is used as input. Included segments are searched on multiple library levels if necessary. The object module generated is saved under the same <longname> as the source module for the main segment.

2.6.4 COPY

The COPY directive specifies that a code segment at a specific level be copied to another segment and level. The names of the "from" and "to" segments may be different.

2.6.5 EXPORT

The EXPORT directive copies a source segment to a sequential file outside the PSL library.

2.6.6 IMPORT

The IMPORT directive copies a source or text segment into a PSL library from an external partitioned data set.

2.6.7 LIBRARY

The LIBRARY directive establishes a library to be used during the current PSL session.

2.6.8 LIST

The LIST directive causes a listing to be produced. The list type can be a segment, module, hierarchy, compile, load, or perform. Source or text segments can be listed in segment and module listings.

2.6.9 LOAD

Using multiple-library level search, external references are resolved by the default linker and the resulting load module and listing are entered into the PSL data base.

2.6.10 MODIFY

The MODIFY directive modifies a segment of source or text. Subcommands specify the editing to be done.

2.6.11 NEWLIB

The NEWLIB directive initializes a new PSL library. The user-ID is subsequently used as the master key for that library.

2.6.12 PERFORM

The PERFORM directive invokes the external tool identified by <external program name>, using the segment identified by <longname> as input. User programs can be executed with the PERFORM.

2.6.13 PURGE

The PURGE directive deletes a segment group from the PSL library. A segment group is all segments with a common longname and level.

2.6.14 QUIT

The QUIT directive allows normal termination of the current PSL session.

2.6.15 REPORT

The REPORT directive produces a management report containing statistical information pertaining to segments modules, programs or library specified.

2.6.16 RESTORE

The RESTORE directive restores a PSL library from a checkpoint file.

2.6.17 SEARCH

The SEARCH directive searches text segments for the specified keyword and string. Where found, it causes a listing of the associated data.

2.6.18 USER

The USER directive sets a default user-ID for a PSL session.

2.6.19 XMIT

The XMIT directive moves a module to a higher library level.

3.0 JOVIAL J73 PSL CONFIGURATION CONTROL CAPABILITIES.

The automated configuration control capability of the J73 PSL will provide the much needed development and maintenance support for JOVIAL J73 programming projects. JOVIAL J73 has been designated by the Air Force as the standard interim programming language for all embedded computers used in avionic systems (including aircraft, missiles, and munitions) until the DOD language, Ada, becomes available.

The following discussion is an approach to the transfer of J73 PSL technology. Here, a definition of configuration control and management will be provided as a foundation for subsequent discussion. Following the definition will be several discussions of J73 PSL capabilities satisfying this definition.

3.1 Configuration Control

Configuration control may be defined as a methodology concerned with procedures for controlling the contents of a software system - a way of monitoring the status of system components, preserving the integrity of released and developing versions of a software system, and controlling the effects of change throughout the system.

3.2 Configuration Integrity

As shown in Figure 3-1, a library may be created for each project with multiple libraries created for multiple projects. The project manager must first create the library database using the NEWLIB command. Each project is then allocated a library in which various forms of data may be stored and manipulated. Seven data types may be created and manipulated at each of seven project completion levels in the library. The default levels may be used or other levels may be defined by the project manager at library set-up time. This ability to create and manipulate a database, define level usages and create code hierarchies represents a set of implicit capabilities provided by the J73 PSL, which allows project management to implement software system control procedures. These software system control procedures

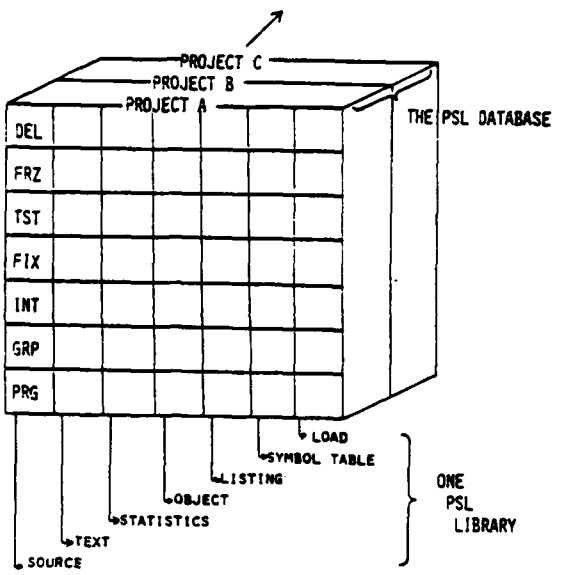


Figure 3-1 J73 PSL Database Structure

will assist the user in maintaining project visibility, software configuration integrity, and system change control.

One J73 PSL procedure for controlling a configuration baseline of a software system is the ability to back up a library on magnetic tape with the CHECKPOINT command. This capability not only provides system control, but also preserves the integrity of released and developing versions of a software system by safeguarding the database against computer failure. The RESTORE command is used to restore a PSL library from a checkpoint file.

The USER command is yet another implicit way of controlling the contents of a software system by preventing unauthorized access to data in libraries. To obtain access to elements of the library, several specific keys must be used, including: the user-ID (a unique user identifier), the longname of the element (a name describing the purpose of the element), the user level, the data type, (main, proc, compool, subr, data, pdl, or other) the version and the language (J73, assembly,

or Programming Design Language (PDL)). The user-ID may be set by the USER command for the entire session while project member is accessing a designated library. When he QUITs the current session and invokes a new LIBRARY, the project member must again declare his unique user-ID. The other access keys are created as a result of the command syntax and then used by the J73 PSL program to verify and restrict further database manipulation.

Another feature of the J73 PSL which is important to configuration control is the ability to maintain version and edition control of any software system. Each time a segment is added or modified in the PSL database, it is given a unique version and edition number. Only the most current edition at a particular level of the library is available for use by a team member. Additionally, management statistics are updated automatically upon segment change to provide the visibility needed for change control.

3.3 Change Control

The preservation of component integrity is an important and valuable aspect of the J73 PSL. The J73 PSL supports integrity in several ways through total system backup procedures, user access restrictions, and automatic code transmission routines.

As previously discussed, the J73 PSL provides control over different versions of programs/systems destruction of data through the use of a back-up capability. The CHECKPOINT and RESTORE commands allow the user to obtain a snapshot of a specified library that can be returned to the system at a later time. The controls implicit in the USER command provide protection of data from unauthorized update. The J73 PSL data protection scheme provides a means to restrict not only access to the levels within a library, but to the data types within a level by restricting users to certain command verbs. Elements of various types within the database that have the same longname and level are defined as a segment group of the library. By specifying the element's longname and level, a user may PURGE all data types

relating to that element, i.e., he can automatically delete the entire segment group of the library. By specifying the element's longname and level, a user may PURGE all data types relating to that element, i.e., he can automatically delete the entire segment group. A user makes use of this capability when creating and then destroying special information for extemporaneous testing and does not wish to keep experimental information in the database for configuration review. Privileged users may also XMIT a module from one level to the next higher level which will automatically promote all other segment groups in the module as well.

The J73 PSL also contains the capability to move a segment group from a higher level to a lower level. If not all elements of a segment exist on the level to which the user wishes to transmit the module, the missing data will be made available at the lower level by automatic drawdown when needed. This facility may be exercised by all users and guarantees that unique configuration baselines will not be violated. (Recall that restrictions based on user-ID and level are enforced such that project team members of lower status cannot violate delivered or tested elements of the library database.)

Another capability provided by the J73 PSL in support of configuration control is the automatic recording and reporting of changes made to the database elements. The vehicle for change effect control of the database elements is the statistics file, in which various data elements are stored for use by the management reporting function. These statistics are available for segment, module, program, and library summary reports by using the REPORT command.

Generally, a request for a summary-by-segment report would specify the project for which the summary is desired. If a project is not specified, separate reports will be generated for all projects in the library. The report may be restricted to segments in a range of library levels. The total number of segment changes is reported as well as the number of segment changes per version. The summary-by-segment report provides project management with an effective way of controlling changes.

The summary-by-program (or module) report is a similar report based on program statistics automatically recorded in the library, and may be obtained for only one or all programs within a range of levels.

The summary-by-library report is useful in that a composite listing of each element of each library level is possible. The CHANGES parameter is indicative of total changes made for all segments at a specified library level and for a specified language.

3.4 Status Monitoring

The ability to monitor the status of system components with the J73 PSL is primarily facilitated by the multi-level management reporting capability described above. Essential to this reporting capability is the seven-level hierarchy representing code completeness. This hierarchy provides a means of formalizing all phases of the software development life cycle from initial coding to software delivery. This concept of the Top Down Structured Programming technology was developed to impose discipline on the traditional way of developing software.

The multi-level management reporting capability is supported by the J73 PSL's capability to maintain an accounting record of statistical data for management report generation. The J73 PSL will automatically collect statistics during execution of the following directives: ADD, MODIFY, COMPILE, and LOAD. A statistics segment is associated with each source segment, each source module, and each load module. The user does not have direct access to statistics segments. Statistics segments are generated automatically and are only readable in the form of a management report obtained through the use of the REPORT command. When using the REPORT command, the user may specify whether statistics from all levels or a range of levels are to be used. As previously described, four types of reports are possible: segment statistic summary, module statistics summary, program statistics summary or library summary. In the first three types, the user may specify a name or receive a summary of all segments, modules, or programs. The library summary consists of a report summarizing the contents of the library in use.

System component status monitoring is also provided through use of the SEARCH command. A user (typically the manager) may SEARCH the PSL database for keywords, which are defined by the manager at library set-up time. Secondary keywords may be used to return a predetermined collection of associated textual material. For example, a searched-for keyword may be PURPOSE (of a programming module) with secondary keywords INPUTS and OUTPUTS. The SEARCH command can be used to scan the library for any or all of these keywords and the associated text accompanying them. Additional information may also be given to narrow the search, such as level or module name. The searched-for textual information may be added to a management status data type and modified for later use or listed out.

4.0 DETAILED DESIGN

The configuration of the J73 PSL includes four independent JOVIAL J73 programs, two FORTRAN programs, and the IBM utility IEBCOPY, as well as external tools, compilers, linkers, and assemblers.

The four independent JOVIAL J73 programs are:

- PSLMAIN - The largest program, this is what is considered The PSL, although the other programs play important roles. Its internal structure will be broken down into compilable modules and explanations of each will be given below.
- PRECOMP - This is a precompiler which processes !COPY statements in JOVIAL J73, FORTRAN, and IBM assembler source modules, producing a temporary sequential source input file for the compilers and assembler. It also processes compool directives for JOVIAL J73 source modules. Use of this precompiler frees the PSL user from having to know DDNAMEs for compool symbol tables and copy files. Also, it makes it unnecessary to change member names in code when it is transmitted to other levels. The COPY processing allows longer modules than 100 lines in each of the three source languages. Precomp also generates some compile statistics.
- PRELINK - Prelink processes a linker control segment to generate INCLUDE or LIBRARY linker control statements prior to linking. The user must place in the linker control source segment the longnames of all modules and compools needed for the link. The use of this prelinker frees the PSL user from having to know DDNAMEs and from having to change the member names in INCLUDE and LIBRARY statements when code is transmitted.
- UPDATE - Update analyzes compiler and linker output to generate values for compile and load statistics.

The two independent FORTRAN programs are ECHO and FECHO, which are needed to reformat the SYSPRINT output from compilers and the IBM linker, respectively, to route this output into the PSL database.

IEBCOPY is used in spawned batch jobs to reformat some compiler outputs for routing into the PSL database. It is also used for backing up a PSL library to tape (CHECKPOINT directive) and for restoring the library to disk (RESTORE). Output listings routed to the printer are generated using IEBCOPY through a spawned batch job originated by the LIST directive. Routing of REPORT output is done the same way.

External tools, compilers, and assemblers invoked in spawned batch jobs through the PSL include:

- the SEA JOVIAL J73 compiler
- the FORTRAN IV Extended H Compiler
- the IBM assembler
- the IBM linker
- the JOCIT JOVIAL J73 compiler
- the J73 Automatic Verification System
- the J73 Code Auditor
- the J73 1750A cross-compiler
- the J73 1750A linker
- the J73 1750A assembler
- the J73 1750A simulator

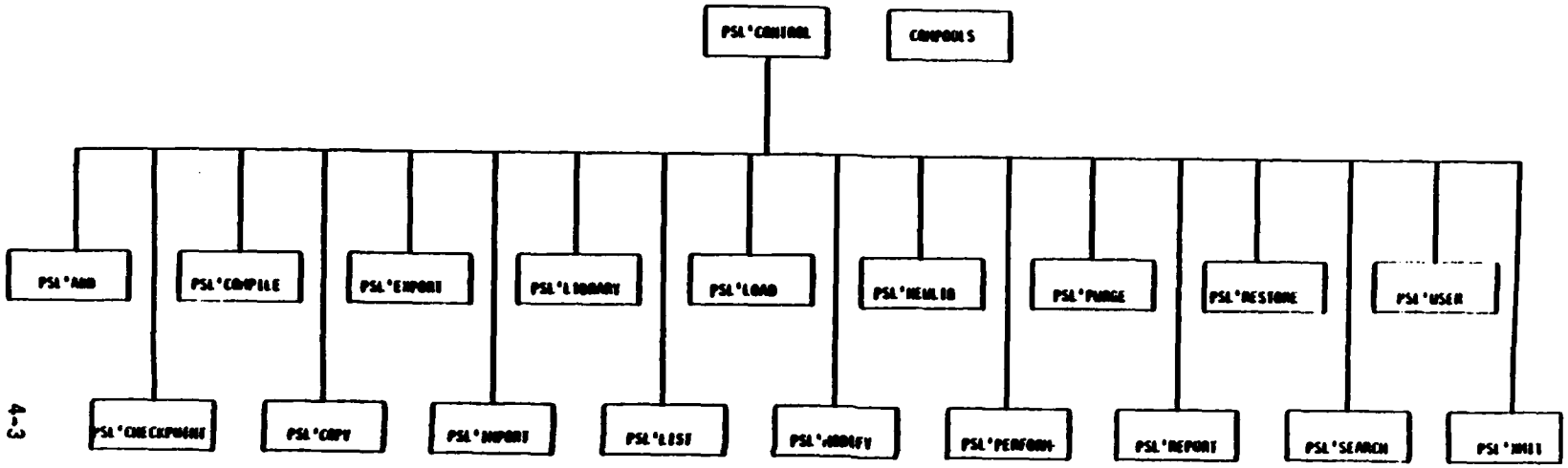
The main PSL program, PSLMAIN, consists of 81 separately compilable modules. These can be broken down by language as follows:

• JOVIAL J73	65	(7 compools)
• FORTRAN	8	
• IBM assembler	8	

The 57 executable JOVIAL J73 source modules can be classified in one of the following catagories:

- Control and major support routines
- Directive execution routines
- Minor support routines

Figure 4-1 shows the hierarchy of the major modules in PSLMAIN.



- PSL*AUTHORIZATION*CHECKER
- PSL*DIRECTIVE*CARD*PARSER
- PSL*MESSAGE
- PSL*DATA*STORAGE*AND*RETRIEVAL
- PSL*SPAWN
- OTHER SERVICE ROUTINES

4-3-562

Figure 4-1. Major Modules in PSLMAIN

Control and major support routines

PSL'CONTROL - This is the top level PROGRAM module of PSLMAIN. It initializes the data structures, controls directive verb parsing, routes control to directive execution routines, and provides error recovery from invalid directive syntax.

PSL'DATA'STORAGE'AND'RETRIEVAL - All input and output, as well as querying of the in-core directory, goes through this control module. It calls the major assembler routine PSL'PERFORM to actually do the I/O operations. Functions provided by PSL'DATA'STORAGE'AND'RETRIEVAL include:

- FIND - locate and read a source or text segment at or above a given level.
- READ - read a source or text segment at a given level.
- WRITE - write or replace a source or text segment at a given level.
- QUERY - locate and read the statistics of a segment with a given longname at or above a given level.
- PURGE - delete all segments of all types having a given longname and level.
- CLOSE - close all open PSL database files.
- FIRST - return the first member name in the in-core directory.
- NEXT - return the next member name in the in-core directory or return end of file.

- PSL'MESSAGE - All joblog messages and prompts are controlled by this routine. It provides facilities for insertion of a string into a message.

- PSL'AUTHORIZATION'CHECKER - Access to levels for reading and writing, access to directive verbs, and access to the database are controlled by this routine. In its initial call it reads the AUTHORIZATIONS source segment to initialize its static tables. PSL'AUTHORIZATIONS also validates the USERID parameter for the USER directive.

- PSL'DIRECTIVE'CARD'PARSER - This is a general directive parser which checks directive syntax against tables indicating types of parameters and whether they are optional or required. The syntax of each parameter type is verified and appropriate error messages are issued.

- PSL'GENERATE'STUBS - This routine processes source code, parsing for the string '!COPY' followed by a longname. It generates a stub for that longname, using for its level the same level as that of the source code it parsed. If a segment group already exists for the COPY longname, a message is issued.

- PSL'GENERATE'HIERARCHY - A recursive routine, PSL'GENERATE' HIERARCHY builds a stack containing all the longnames, shortnames, levels, and copy nesting depths for segments in a module. It is used in the LIST, REPORT, and SEARCH directives, as well as in the precompiler.

- PSL'SPAWN - Spawn generates batch background jobs for the following directives: COMPILE, LOAD, PERFORM, LIST, REPORT, CHECKPOINT, and RESTORE. It looks for certain strings in JCL templates and substitutes values from a global table, placed there by the caller. The resulting JCL file is written to the internal reader (for batch execution) or to a file (for interactive execution).

Directive Execution Routines

All directive execution routines call PSL'DIRECTIVE'CARD'PARSER to check syntax. Then, if applicable, they call PSL'AUTHORIZATION' CHECKER to check access. Semantics is then checked, and if there is no conflict, the directive is carried out. If there is an error at any point in the syntax, access, or semantics checking, the directive processing is terminated. Some message indicating affirmative action or denial of the directive will always be given.

A very brief description of each directive execution routine is given below:

- PSL'ADD - Semantics checks include:
  - For a top level ADD, unique shortname.
  - For a non top level ADD, stub must exist, language must match for source, type must be PROC or LOCAL for source.

  For a top level ADD, a segment group is created in the database consisting of SOURCE, TEXT, and STATISTICS segments. For a non-top level ADD, the stub segment is replaced. Text following the directive is checked for size and parsed for !COPYs. Statistics are initialized. Stubs are generated for !COPYs.

- PSL'CHECKPOINT - A job is generated to unload the currently used PSL library to the indicated tape.

- PSL'COMPILE - A job is generated to compile or assemble the source module. The language of the source module determines which compiler or assembler is used. Options from the directive are placed in the generated job. Semantics checks include finding the module's top level segment on the given level, with the language JOVIAL, ASM, or FORTRAN.

- PSL'COPY - Semantics checks include:
  - to-level less than or equal to from-level.
  - to-longname does not exist on the to-level.

  A copy of the source, text and statistics segments is made at the to-level. If the longname is not changed, only the level will be changed in the stats. If the longname is changed, stats are reinitialized.

- PSL'IMPORT - Semantics checks include:
  - If not a REPLACE, shortname must be unique.
  - For a REPLACE, segment being replaced must exist on the level given.
  - If the member name of the source data segment in the external PDS is not specified, one is generated as it would be in the PSL. The segment in the external PDS must exist.

The IMPORTed source data is parsed for COPYs and stubs are generated, as in PSL'ADD. Statistics are initialized. For a REPLACE, statistics are reinitialized. For a REPLACE, only the segment IMPORTed is changed. No change is made in the TEXT segment for the REPLACE of a source segment. If the IMPORT is not a REPLACE, a new source, text, and stats segment group is created, as in PSL'ADD.

- PSL'EXPORT - Semantics checks include finding the segment to be exported on the given level. If no filename for the external member is given, one is generated. The source or text segment is copied into the external PDS member.

- PSL'LIBRARY - The libname parameter is used to override that in the profile, but the libname parameter must agree with the name used with files ALLOCATEd or used in DDNAMEs, for interactive and batch execution, respectively. AUTHORIZATIONS and KEYWORDS segments are accessed to set values in internal tables.

- PSL'LIST - This directive provides six kinds of listings: a single source or text segment, all the source or text segments in a module, the copy hierarchy of a module, or one of the output listings (compile, link, or perform). Five subdirectives (#HEADER, #NONAME, #NUMBERING, #SPACING, #LINES) provide format control. Routing to the printer is possible. Routed listings result in a spawned batch job. FORTRAN and IBM assembler routines are used for I/O.

- PSL'LOAD - Semantics checks include verifying the existence of the top level source segment at the given level and the existence of the source segment for link control input, of language LINK at the same level. A job is generated to invoke the linker with the run-time library appropriate to the language of the source module.

- PSL'MODIFY - Semantics check includes verifying the existence of the segment at the given level. The segment is read into a buffer. Five subdirectives (#INSERT, #DELETE, #CHANGE, #LIST, and #END) may be used to edit the segment. Execution of the segment in the database. Statistics are updated.

- PSL'NEWLIB - This directive initializes an empty library that has been created using system procedures. Source and text stubs for AUTHORIZATIONS and KEYWORDS are created. The userid parameter is used as the master key and stored in the statistics segment for AUTHORIZATIONS. PSL'NEWLIB calls PSL'AUTHORIZATION' CHECKER to initialize the internal tables, but there is no data in the AUTHORIZATIONS source segment at this time, so only the master key is authorized for this session. Similarly, the KEY-WORDS segment is still a stub, so the default level names and no keywords are in force.

- PSL'PERFORM'DIRECTIVE - Semantics checks are similar to those for PSL'COMPILE and PSL'LOAD. A job is spawned to execute the indicated external tool or to execute the user's load module (external-tool-type EXECUTE). Output may be routed into the PSL (the default) or to the PRINTER and external files. The PSL'PERFORM module is supported by 9 other modules: PSL'EXECUTE, PSL'LINK'JOCIT, PSL'JOCIT'COMPILER, PSL'1750A' COMPILER, PSL'JAVS, PSL'J73'CODE'AUDITOR, PSL'1750A'LINK, PSL'1750A'SIMULATOR, and PSL'1750A'ASSEMBLER. These modules define values for substitution into the templates.

- PSL'PURGE - This routine calls PSL'DATA'STORAGE'AND'RETRIEVAL to delete all segments with a given member name (derived from given longname and level).

- QUIT - Quit terminates a PSL user session and returns to the TSO mode.

- PSL'REPORT - The report routine generates four types of reports:
  SEGMENT - individual segment statistics are displayed and summarized
  MODULE - compile statistics are displayed and summarized
  PROGRAM - load statistics are displayed and summarized
  LIBRARY - overall PSL library statistics are generated and displayed in several tables

  The routine makes use of two optional level parameters and a parameter which can either be the longname of a module, or a four-character CPCG. These parameters can limit the range of the reports.
  Reports can be routed to the printer by spawning a batch job to copy the generated output.

- PSL'RESTORE - Restore generates a batch job to copy the backed-up PDS's of a CHECKPOINTed PSL library from tape to disk. The name must be changed (unless the old PSL library was deleted, which is dangerous), and the master key can be changed.

- PSL'SEARCH - Search makes use of the keywords in TEXT segments to print selected information from the TEXT segments. Parameters of the directive control the search algorithm and the material to be printed.

- PSL'USER - User sets the default userid and prints a message telling what the current default is. If desired, only the message can be printed, with no change of the default.

- PSL'XMIT - Semantics checks include existence of top level segment on from level and to-level greater than from-level. Write access at to-level is required. A module PSL'XMIT'SEGMENT is called to perform the copy and purge operation.

Minor Support Routines

- PSL'FLUSH'INPUT'CARD - Used to recover from directive aborts.

- PSL'SCAN'ASCII'STRING - Scanner for directive input, returns tokens and delimiters, keeps track of scan of record.

- PSL'FETCH'DIRECTIVE'INPUT - Returns a directive verb from an input record.
- PSL'FIND'ASCII'KEYWORD - Scanner which searches directive verb list.
- PSL'CONVERT'HEX'BEAD - Converts hexadecimal alphanumeric characters to binary numbers. Used when reading function masks in PSL'AUTHORIZATION'CHECKER.
- PSL'EXTERNAL'LONGNAME'VALID - Checks syntax of PDS member names.
- PSL'CARD'BUFFER - Interface for obtaining contents of three source buffers.
- PSL'DATE - Returns date.
- PSL'TIME - Returns time.
- PSL'KEYWORD'INDEX - Returns subscript of keyword in table.
- PSL'CONVERT'CHAR'TO'DGT - Converts character to digit for numbers 0 to 9999999.
- PSL'CONVERT'CHAR'TO'DGT1 - Same as above, but not in a recursive loop with PSL'MESSAGE.
- PSL'ASSIGN'BFR - Interface for assigning to three buffers.
- PSL'CREATE'LIST'MEMBER - Creates member names.
- PSL'READ'INPUT'CARD - Gets a record of directive input.
- PSL'CONVERT'DIGIT'TO'CHAR - Converts numbers to characters for insertion into messages.
- PSL'CONVERT'DGT'TO'CHAR - Converts a decimal number to a character string and returns the length of the string.
- PSL'CONVERT'HEX'CHAR'STRING - Converts a hex character string to a decimal integer.
- PSL'XMIT'SEG - moves a segment group from one level to a higher level.

- PSL'JOCIT'COMPILER - modifies JOCIT compile template.
- PSL'CREATE'LIST'MEMBER - generates a member name for a member of the LIST PDS.
- PSL'EXECUTE - modifies EXECUTE template.
- PSL'LINK'JOCIT - modifies JOCIT link template.
- PSL'1750A'COMPILER - would modify 1750A compiler template.
- PSL'J73AVS - would modify J73 AVS tamplate.
- PSL'J73'CODE'AUDITOR - would modify J73 code auditor template.
- PSL'1750A'LINK - would modify 1750A linker template.
- PSL'1750A'SIMULATOR - would modify 1750 simulator template.
- PSL'1750A'ASSEMBLER - would modify 1750A assembler template.
- UPDATE'SCANNER - examines output listings for UPDATE.
- PSL'SCANNER - same as PSL'SCAN'ASCII'STRING except no message #213.
- PSL'TEMP'CARD'BUFFER - function returns contents of three temporary buffers.

FORTRAN modules

INSTR - Reads an 80-column record.

OUTSTR - Writes an 80-column record, with carriage control.

TIME - Returns time and date.

WRTPCL - Writes an 80-column record, without carriage control.

RWIND - Rewinds a device.

OUTF - Writes a 133-column record, with carriage control.

RDLIST - Reads a 133-column record. (Part of UPDATE.)

MDLIST - Write an 80-column source record with a line number.

WRTSP - Writes an 80-column record for PSL'SPAWN.

ECHO - Copies SYSPRINT output to PSL LIST PDS. (Independent program.)

FECHO - Copies PSL LIST PDS Segment to printer. (Independent program.)

ASSEMBLER modules

PSL'PERFORM - Does all I/O for the PSL database. Performs the functions that PSL'DATA'STORAGE'AND'RETRIEVAL provides.

PSL'BUILD'DIRECTORY - Builds an in-core directory by reading the directory of the STATS PDS.

PSL'EXTERNAL'FILE'INTERFACE - Performs I/O for external PDS for IMPORT and EXPORT.

SYSTIM - Generates time and date.

ASCII - Converts EBCDIC to ASCII.

EBCDIC - Converts ASCII to EBCDIC.

PSL'READ'LISTING - Reads a 30 record block of 133-column records.

PSL'READ'TEMPLATE - Read a JCL template.

5.0 TESTING

Code was tested in three phases: unit, integration, and qualification. The same criteria were used for each phase. Both legal and illegal inputs were tested. Semantics were tested, and the full range of parameters were used to test boundary conditions. Attempts were made to trace execution through all major branches of the code.

Unit testing proceeded throughout the PSL development. Individual modules were unit tested, either with drivers and stubs, or with the top level acting as a top-down driver. As stubs were replaced by tested routines, the use of stubs diminished. This was especially true when the major service routines, such as PSL'DATA'STORAGE'AND'RETRIEVAL and PSL'AUTHORIZATION'CHECKER were completed. Unit testing then shaded into integration testing.

Integration testing involved executing the main PSL program as a whole, with undeveloped portions stubbed out. In addition to the criteria discussed above for unit testing, integration testing also tested interfaces between modules. Misinterpretation of inputs and outputs to modules, which is not apparent during unit testing, was immediately discovered during this phase of the testing. Whenever a new module or modules were introduced into the load module used for integration testing, tests were repeated to insure that the new code did not degrade the performance of the PSL.

Concurrency tests were carried out to explore the possibility of contention. If was decided that few or no problems at all with concurrency exist. Attempts were made to create a problem by having parallel PSL sessions operate on the same segments concurrently, but no difficulties came to light. However, during the training class, it was discovered that ADDS in parallel sessions could override each other.

Qualification testing took place during the week of October 26-30, 1981. The Test Plan, completed in September 1981, served as a basis for the qulaification tests. Again, the types of tests fall into four catagories: legal inputs, illegal inputs, correct execution (semantics tests), and capacity and constraint tests. The tests were derived from the Statement of Work. Testing techniques ranged from visual inspection of code and documentation to batch and interactive operation, parallel sessions, and use of external tools (text editor, etc.) to provide the test results.

6.0 USER INDOCTRINATION

A training class for twenty persons was held at ASD on October 20-24, 1981. During the first two days an extensive set of viewgraphs and a tutorial made from the viewgraphs were used as a basis for lectures. The lecture topics were:

- What is a Program Support Library?
- Configuration Management and Control.
- J73 PSL Definitions.
- The J73 PSL Database.
- Introduction to directives; general syntax and operation.
- Detailed discussion of the 19 directives.
- Control of the PSL; AUTHORIZATIONS and KEYWORDS.
- Usage conventions.

The third and fourth day were primarily laboratory, in which the attendees used the PSL on six terminals at ASD. The objectives of the third day in the laboratory were to generate source code, compile, link, and execute it within a pre-existing PSL library. Directives used during this program development phase included: ADD, COMPILE, COPY, EXPORT, IMPORT, LIBRARY, LIST, LOAD, MODIFY, PERFORM, PURGE, QUIT, and USER. The last day in laboratory emphasized the management aspects of the PSL. The following directives were

either used by those in the class, or demonstrated, in cases where it would have been impractical for each person to use them: NEWLIB, CHECKPOINT, RESTORE, XMIT, SEARCH, and REPORT. Setting up the TSO procedures, files, and JCL to implement tne PSL were also discussed on this last day.

Forms were provided on which suggestions for improvement of the PSL were requested.

7.0 SUGGESTED ENHANCEMENTS

This is a list of possible enhancements to the JOVIAL J73 Programming Support Library. The enchancements are divided into four catagories:

- major enhancements, involving new design;
- minor enhancements which can be implemented easily with the present design;
- features already designed but unimplemented due to lack of hours under the development contract;
- important bugs in the present PSL which will require major revision in order to be fixed.

Each category is elaborated below.

7.1 Major Enhancements Requiring New Design

7.1.1 HELP/Tutorial facility. A group of segments in the user's PSL library could contain PSL syntax and semantics. However, a more effective design would involve a series of menus, permitting the user to quickly locate the needed information. The menu approach can also serve as a tutorial for persons unfamiliar with the PSL.

7.1.2 Traceability Reports. A report could be generated which would map requirements to code. Text files would be accessed, as well as statistics segments.

7.1.3 Software Problem Reports. SPR's could be generated, using user input, statistics, and other PSL-generated information. Prompts for an interactive user can facilitate filing complete reports. The existing authorization and configuration controls will apply.

7.1.4 Directory Command. This command is very much needed so the user can keep track of longnames of segments. It could be implemented as an option of the REPORT directive to take advantage of that directive's discrimination parameters for level, longname, TEXT, userid, and CPCG. However, this might interfere with effective authorization control with respect to the REPORT directive. Perhaps a new directive should be designed for this function.

7.1.5 Edit Capability for Listings. It would be helpful to be able to search for compiler error messages in compile listings, for example.

7.1.6 Reduction of Load Module Size. One possible savings in load module size might be to eliminate the redundant copy of PSL longname defines which occurs in every compilable module of the PSL.

7.1.7 An Enhanced Text Editor. Some or all of the following features might be added to the existing MODIFY directive:

a. numbering which is not automatic and consecutive integers.
b. redesign the CHANGE subdirective, permitting change of a single occurrence of a string in a line of text.
c. MOVE subdirective.
d. COPY subdirective.
e. control of verification echoes.
f. an ALTER subdirective which permits insertion or change of characters by position rather than string replacement.

7.1.8 Improved Tool Interfaces. External files could be employed in some applications. For example, external files could be used as input to the EXECUTE function of the PERFORM directive. Alternatively, EXECUTE input segments could be redesigned to permit !COPY segments, to get around the limit of 100 lines imposed by the PSL.

7.1.9 Data Protection. Investigate the possibility of data protection using system functions.

7.1.10 Direct Use of System Editors. Investigate the possibility of executing TSO Edit and Clemson editors directly, without exit from the PSL, by using the assembler LINK macro.

7.1.11 Direct Interactive Job Submission. Investigate the possibility of submitting batch jobs interactively without exiting the PSL. (TSO Edit and Clemson editor can do this.)

7.2 Minor Enhancements Which Can Be Implemented Under the Present Design.

7.2.1 Improved Messages.

7.2.2 Stand-Alone FORTRAN and ASSEMBLER Batch Execution. Presently, these languages can only be used to provide support procedures for J73. Additional link templates would be needed to execute a FORTRAN or assembler program on its own.

7.2.3 Abbreviations for Commands and Parameters.

7.2.4 Lower Overhead for Entry to the PSL. Due to the necessity of exiting the PSL to submit batch jobs during an interactive session, one must repeatedly enter the LIBRARY and USER directives. Even if a way is found to avoid having to exit the PSL to submit jobs, it will still be necessary to exit the PSL to examine JCL output from those batch jobs. A default scheme for the LIBRARY and USER directives, and perhaps level as well, would be desirable. This enhancement may not be quite so minor.

7.3 Unimplemented Features.

7.3.1 Changing the Master Key with the RESTORE Directive.

7.3.2 Make the Link Control Segment in LOAD Optional.

7.3.3 Implement the JAVS Interface.

7.3.4 Implement the 1750A Tools Interfaces.

7.3.5 Implement the Code Auditor Interface.

7.3.6 Route Results of REPORT and LIST Directives to the Printer.

7.3.7 Implement the EXTRN Routing Parameter in PERFORM.

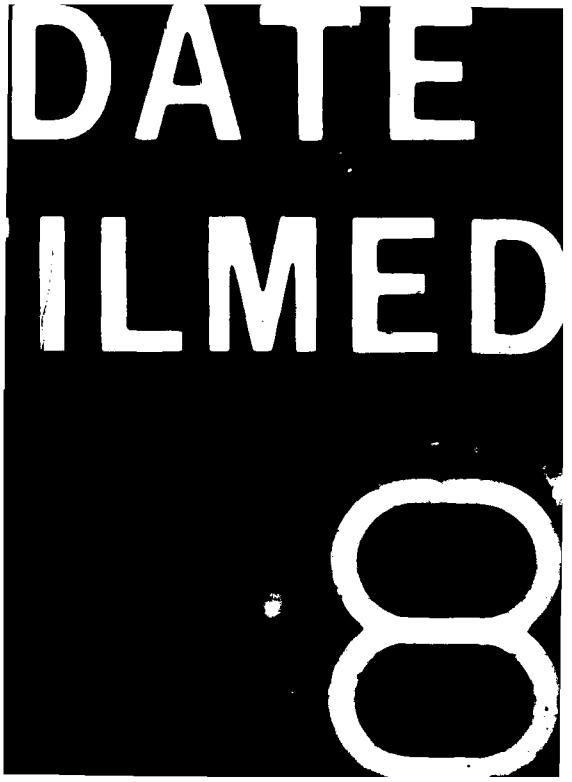
7.3.8 Route JOCIT Compile Output into the PSL Database.

7.4 Persistent Bugs.

7.4.1 Parallel ADD. During the training class, it was discovered that the PSL design did not reinitialize the in-core directory. Thus, persons using the same PSL library in parallel sessions crash each other's segments when they use the ADD directive. The solution to the problem involves assembler programming.

7.4.2 PSL Library Capacity. Although the design calls for a capacity of 1000 segments in a PSL library, the implementation seems to permit only about 225. There may be a simple solution, involving a change to one parameter, but the real problem is not yet understood.

7.4.3 Formatting Problem in COMPILE, PERFORM, and LOAD Listings. These listings currently do not have the correct appearance, suggesting that there is an incompatible record length in one of the PSL Modules.


DATE
ILMED
8